Fr. Conceicao Rodrigues College of Engineering

Department of Computer Engineering

Academic Term : July-Nov 2023-24

Class : T.E. (Computer B)
Subject Name: Computer network Lab
Subject Code : CSL 502

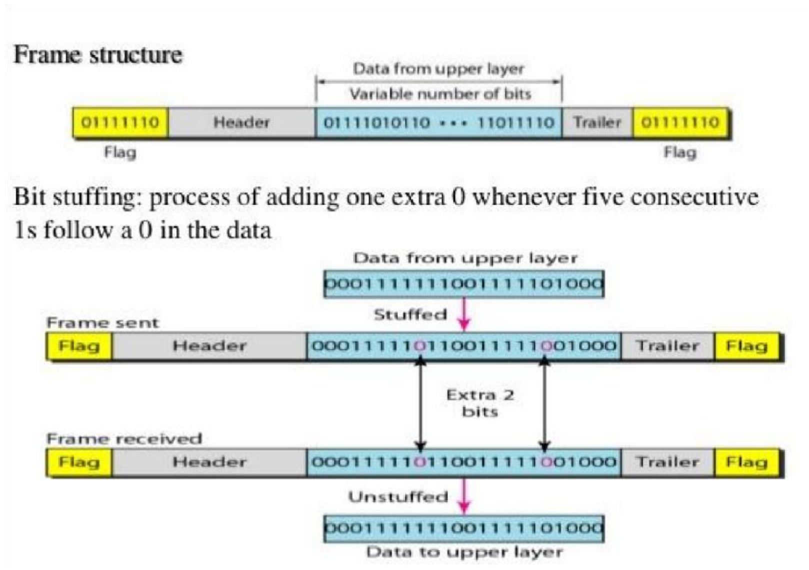
Experiment No:	3
Date of Performance:	8/8/23
Roll No:	9542
Name of the Student:	Chris Gracias

AIM: Implementation of bit stuffing and unstuffing algorithm

Theoretical description:

To avoid the appearance of flag pattern in the data field of the frame; thereby avoiding undue termination of the frame and loss of data bit stuffing and destuffing is used.In HDLC protocol, the flag field delimits the frame at both ends with the unique pattern 01111110. The pattern 01111110 may appear somewhere inside the frame. It may destroy the synchronization. To avoid this problem, a procedure known as bit stuffing is used. For all bits between the starting and ending flags, the transmitter inserts an extra 0 bit after occurrence of five 1s in the frame.

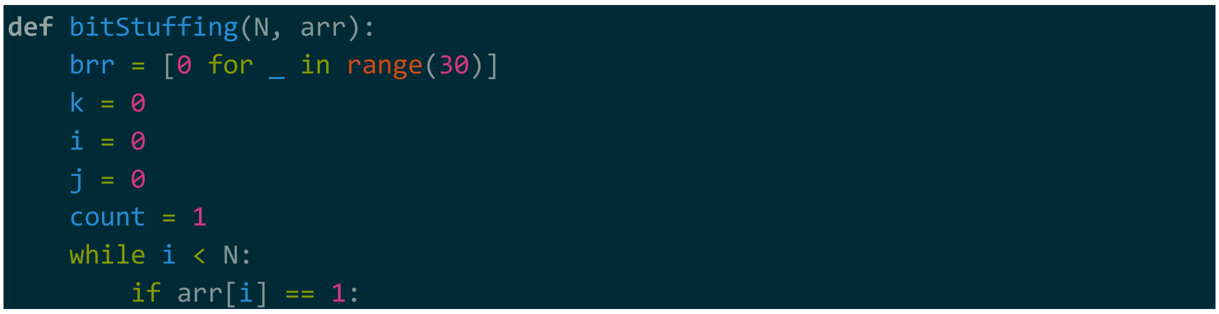
After detecting a starting flag, the receiver monitors the bit stream. When the pattern of five 1s appears, the sixth bit is examined. If this bit is 0, it is deleted. If the sixth bit is 1 and the seventh bit is 0, the combination is accepted as a flag. Thus, with the use of bit stuffing, arbitrary bit patterns can be inserted into the data field of the frame. Figure below shows an example of bit stuffing and unstuffing.



Bit stuffing: process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data

Figure 1: Example of bit stuffing and unstuffing

Write down program to implement bit stuffing and unstuffing using C/C++/Java/Python and Attach the screenshot of Program and its output



```
def bitStuffing(N, arr):
    brr = [0 for _ in range(30)]
    k = 0
    i = 0
    j = 0
    count = 1
    while i < N:
        if arr[i] == 1:
```

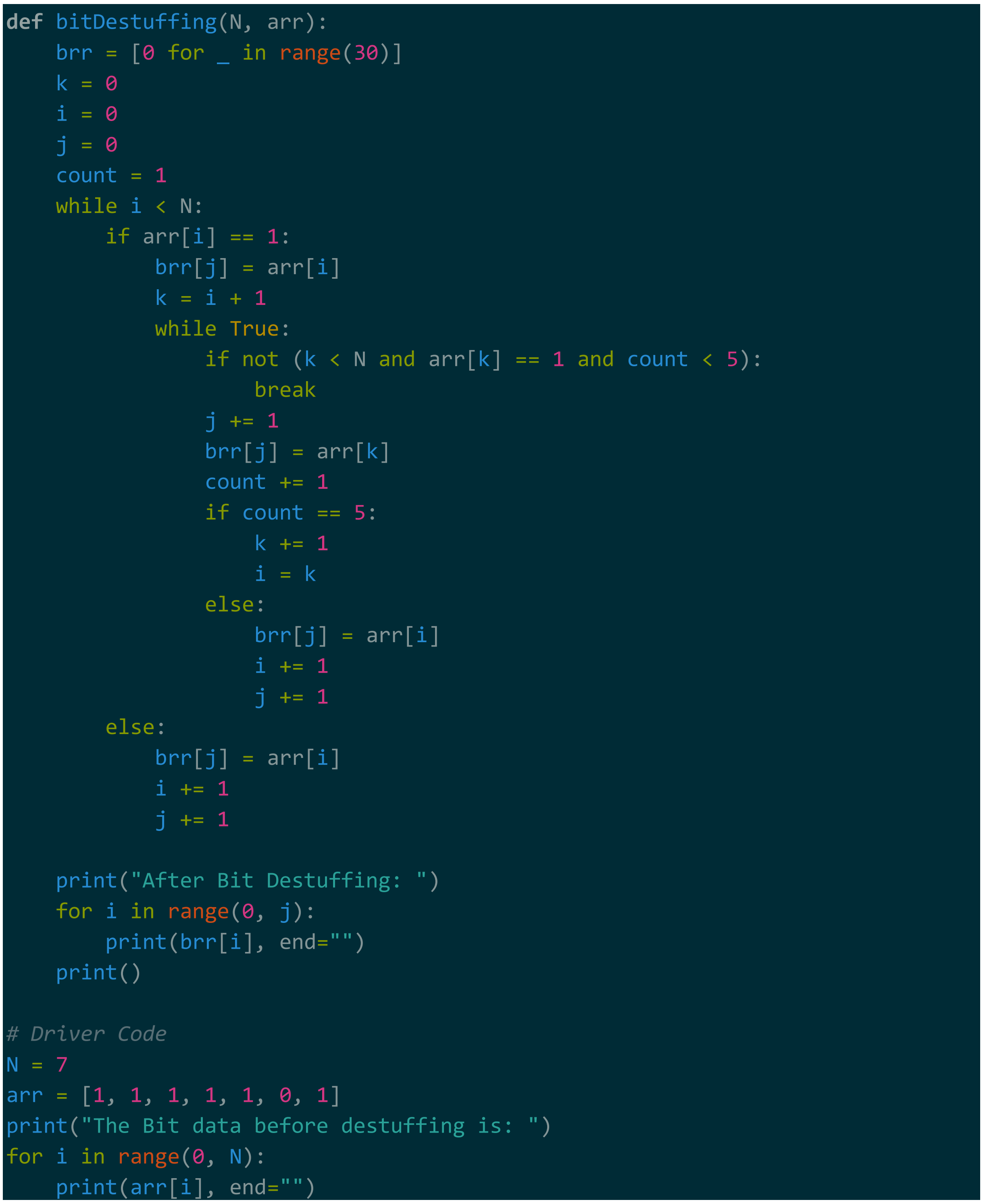
```python
            brr[j] = arr[i]
            k = i + 1
            while True:
                if not (k < N and arr[k] == 1 and count < 5):
                    break
                j += 1
                brr[j] = arr[k]
                count += 1
                if count == 5:
                    j += 1
                    brr[j] = 0
                    i = k
                    k += 1
                else:
                    brr[j] = arr[i]
                    i += 1
                    j += 1

    print("After Bit Stuffing: ")
    for i in range(0, j):
        print(brr[i], end="")
    print()

# Driver Code
N = 6
arr = [1, 1, 1, 1, 1, 1]
print("The Bit data before stuffing is: ")
for i in range(0, N):
    print(arr[i], end="")
print()
bitStuffing(N, arr)
```

Output:

```
The Bit data before stuffing is: 111111
After Bit Stuffing: 11111011
```

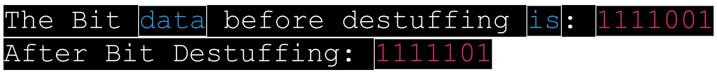


```python
def bitDestuffing(N, arr):
    brr = [0 for _ in range(30)]
    k = 0
    i = 0
    j = 0
    count = 1
    while i < N:
        if arr[i] == 1:
            brr[j] = arr[i]
            k = i + 1
            while True:
                if not (k < N and arr[k] == 1 and count < 5):
                    break
                j += 1
                brr[j] = arr[k]
                count += 1
                if count == 5:
                    k += 1
                    i = k
            else:
                brr[j] = arr[i]
                i += 1
                j += 1
        else:
            brr[j] = arr[i]
            i += 1
            j += 1

    print("After Bit Destuffing: ")
    for i in range(0, j):
        print(brr[i], end="")
    print()

# Driver Code
N = 7
arr = [1, 1, 1, 1, 1, 0, 1]
print("The Bit data before destuffing is: ")
for i in range(0, N):
    print(arr[i], end="")
print()
bitDestuffing(N, arr)
```

Output:



```
The Bit data before destuffing is: 1111001
After Bit Destuffing: 1111101
```

Conclusion: Successfully implemented of bit stuffing and unstuffing algorithm

Post lab questions:

1. Explain HDLC protocol of data link later in detail

HDLC is a synchronous communication protocol operating at the data link layer of the OSI model. It offers reliable data transmission with framing, addressing, and error-checking. Key features include:

Framing: HDLC encapsulates data with start and end flags, address, control, and error-checking.
Modes: ABM supports symmetric communication, ARM has initiator-responder roles.
Frame Types: I-Frames (data), S-Frames (flow control, error checking), U-Frames (control).
Flow Control: Manages data transmission rate to prevent overload.
Error Detection: Uses CRC for error detection and correction.
MLP: Bundles channels for increased bandwidth (Multilink Protocol).
Frame Format:

Flag (Start delimiter)
Address (Optional)
Control
Data

FCS (Frame Check Sequence)
Flag (End delimiter)
Modes:

NRM: Initiator-responder, simplex.
ABM: Symmetric full-duplex.
ARM: Initiator-responder, simplex with responder initiation.
HDLC underpins PPP and LAP-B, contributing to reliable data communication across various networks.

2.Differentiate between CSMA/CD and CSMA/CA

CSMA/CD (Carrier Sense Multiple Access with Collision Detection):

Usage: Primarily used in wired networks, like Ethernet.
Collision Handling: Detects collisions during transmission and stops sending data, then waits for a random time before retransmitting.
Collision Detection: Monitors the network to detect collisions while transmitting.
Efficiency: Less efficient due to the time taken to detect and handle collisions.
Wireless: Not suitable for wireless networks due to hidden node problem.

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance):

Usage: Used in wireless networks, like Wi-Fi.
Collision Handling: Focuses on collision avoidance rather than detection. Uses techniques to minimize collisions.
Collision Detection: Less emphasis on collision detection, as avoiding collisions is the priority.
Efficiency: More efficient in wireless networks, reducing the chances of collisions.
Wireless: Designed for wireless networks, addresses the hidden node problem through RTS/CTS (Request to Send / Clear to Send) mechanisms.
In summary, CSMA/CD is suitable for wired networks and relies on collision detection and subsequent retransmission, while CSMA/CA is designed for wireless networks and prioritizes collision avoidance to improve efficiency and address the hidden node problem.